This project wil be due before the end of class either Monday May 15 or Wednesday May 17. You will demonstrate it to me. This project will be worth 30% of your grade.

What Do You Have to Do?

1. Create classes and data retrieval/maintenance methods for the Reviews and Members tables. I will provide you with the classes for the Film related tables(Films, Categories & FilmCategories).
2. I will provide you with the code to implement the film CRUD methods.
3. Create a suitable master page and dropdown menu
4. Implement the required pages outlined in this document(FilmList, ViewFilm, Maintain Members, MyProfile and Write a Review).
5. We probably will not get to the security portion of the project.

Web Based Project Management

Regardless of which development methodology we decide to use there are things we need to do before we can do any coding. Here are my ideas on what we need to know before we start coding the final site.

1. What is the purpose of the website?
2. Who will be using the site?
3. How will they be using it?
4. What kind of security does our site need?
5. Gathering of images, data and sample pages that may be used in the final site.
6. A preliminary database Entity Relationship Diagram(ERD).

Preliminary Design

Here are some answers to the above questions:

1. **Purpose of website**
   The Lakeside Art Film Group has met for a couple of years at various sites to view films/dvds related to art. Some of these films are rebroadcasts of PBS shows and almost all of them are documentary in nature. After a communal viewing of a film, the group has a general discussion of the work as well as any comments on the subject matter. The group would like to formally record the reviews of the members and have a central place where information on all the films can be found. They also wish to show additional resources members can use to further study the film.

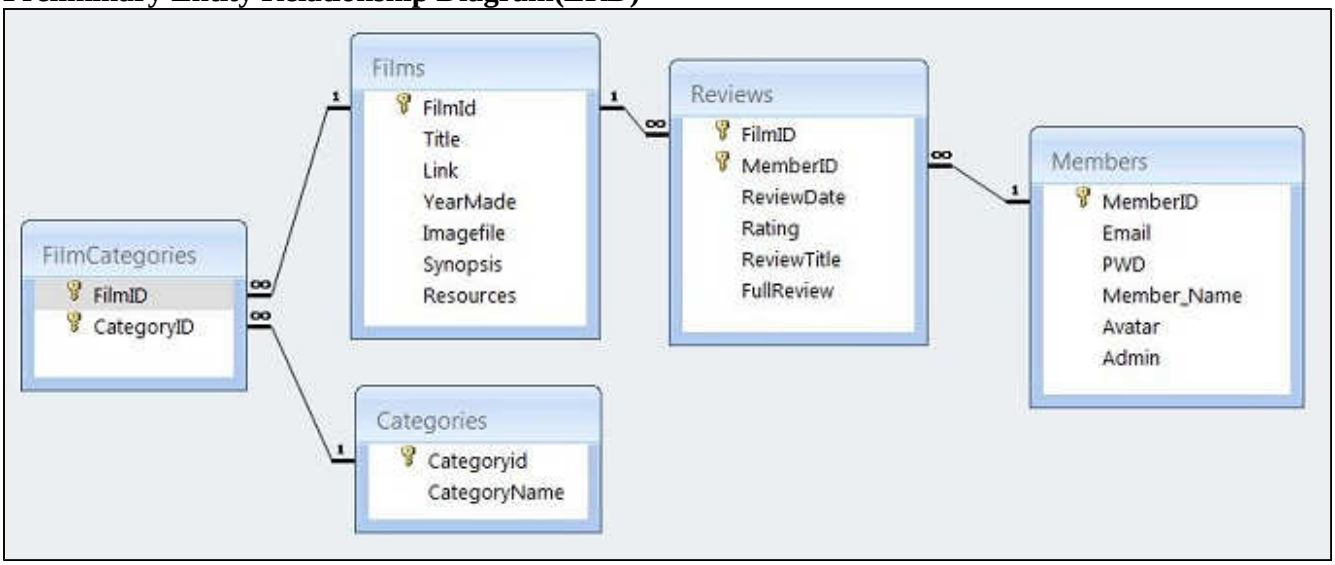
2. **Who will be using the site?**
   Anyone can view the film and review data but only members of the group can post their own reviews. The group is very democratic and anyone can join. One person in the group is designated as the administrator and he/she will setup a new member as well as post/update the film data.

3. **How will people use the website?**
   Anyone can view the film/review data but only members can post reviews. A login screen will be used to authenticate members.

4. **What kind of security does our site need?**
   We will use forms authentication so that a user must present their id and password before they can post a review. Role based security will be used and authenticated users will be either in the admin or member role. Due to the small and select group, we are not going to be too concerned about SQL injection attacks or cross site scripting. We also need to allow a wide range of data to be inputted to support the film biodata as well as review data.

5. **Do we have any sample data/images?**
   The Internet Movie Database is an excellent spot to get a photo of the dvd and as well as most of the bio data on a film. There are reviews on this site whose format we can copy as well as Rotten Tomatoes for review examples. Sample avatar images can be found online and users can make up their own using a paint software program.

6. **Preliminary Entity Relationship Diagram(ERD)**



The group recognizes that there are many good sites that contain excellent biographic data about each film(PBS and Internet Movie Database). They do not wish this site to compete with either one which is why the amount of data recorded for each film seems small. The synopsis and resources fields are of a memo type and can store a tremendous amount of data that will be of a freeform nature. We need to handle carriage return/line feed combinations and allow for almost any input including html tags in the Resources field in the Films table.

General Help

1. **Global Database Connection Object**
   We can use this global dbcon statement to create the connection object:

```
1 SqlConnection dbcon = new SqlConnection(ConfigurationManager.
2           ConnectionStrings["lakesidedb"].ConnectionString.ToString());
```

   The variable will be created when the controller module is started.

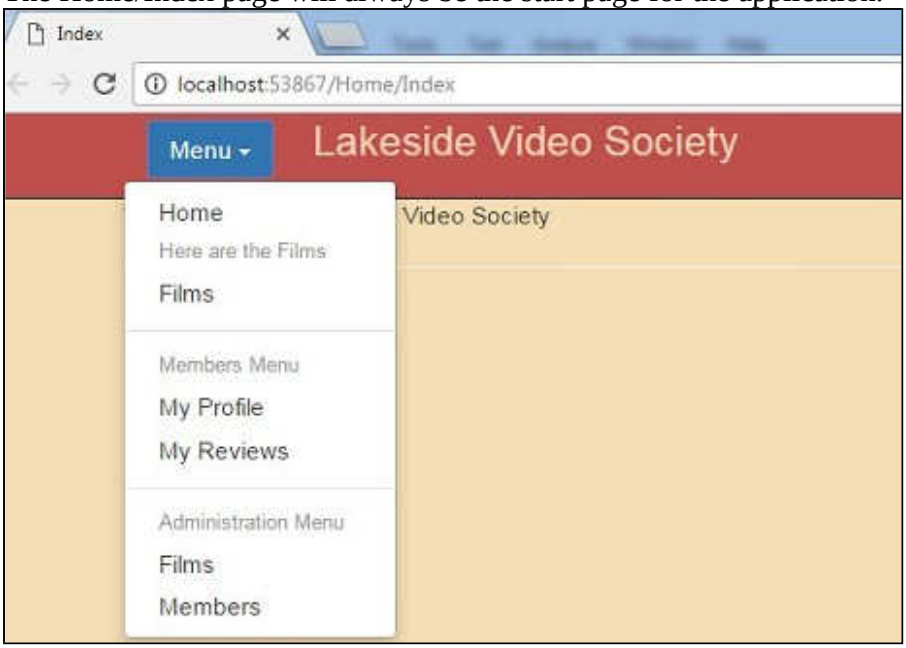
2. **Beware max-width in Site.css file**
   There is a directive that limits the maximum width of all input related tags to 280px, You can override this on each individual tag by using this inline style statement `style="max-width:xxxpx;width:xxxpx"`.

3. **Mind your Cache!**
   This seems to be more a problem with Chrome than with the other 2 browsers. To do it in Chrome you can issue the Ctrl-Shift-Delete command which will bring up the Clear browsing data dialog box. Choose the past hour or day option and then click the Clear browing data button.

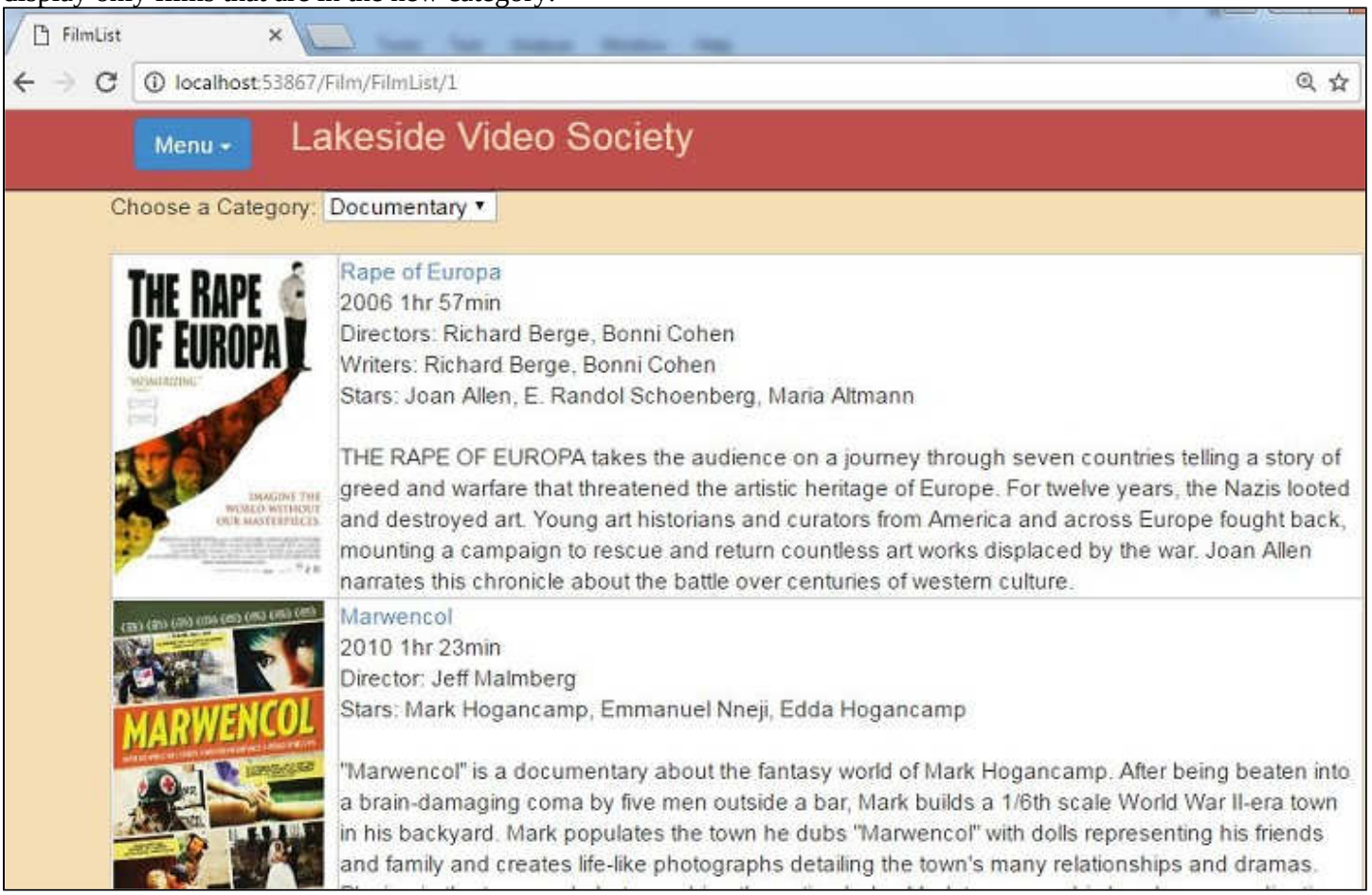
4. **Do not try and Run with Serious Errors!**
   It is tempting to run your project with in spite of serious errors. many times this may work but there are times when serious debugging problems may result. Comment out code until a clean compile is achieved.

Required Pages

1. **MasterPage and Home**
   You need to create a MasterPage that will contain the name of the Film Society along with a navigational menu. The Home/Index page will always be the start page for the application.



2. **Filmlist**
Clicking on the View Films menu link will bring up this page. This page presents a dropdownlist showing all the categories. The first category is preselected but the user can change the selected value in the dropdownlist and display only films that are in the new category.

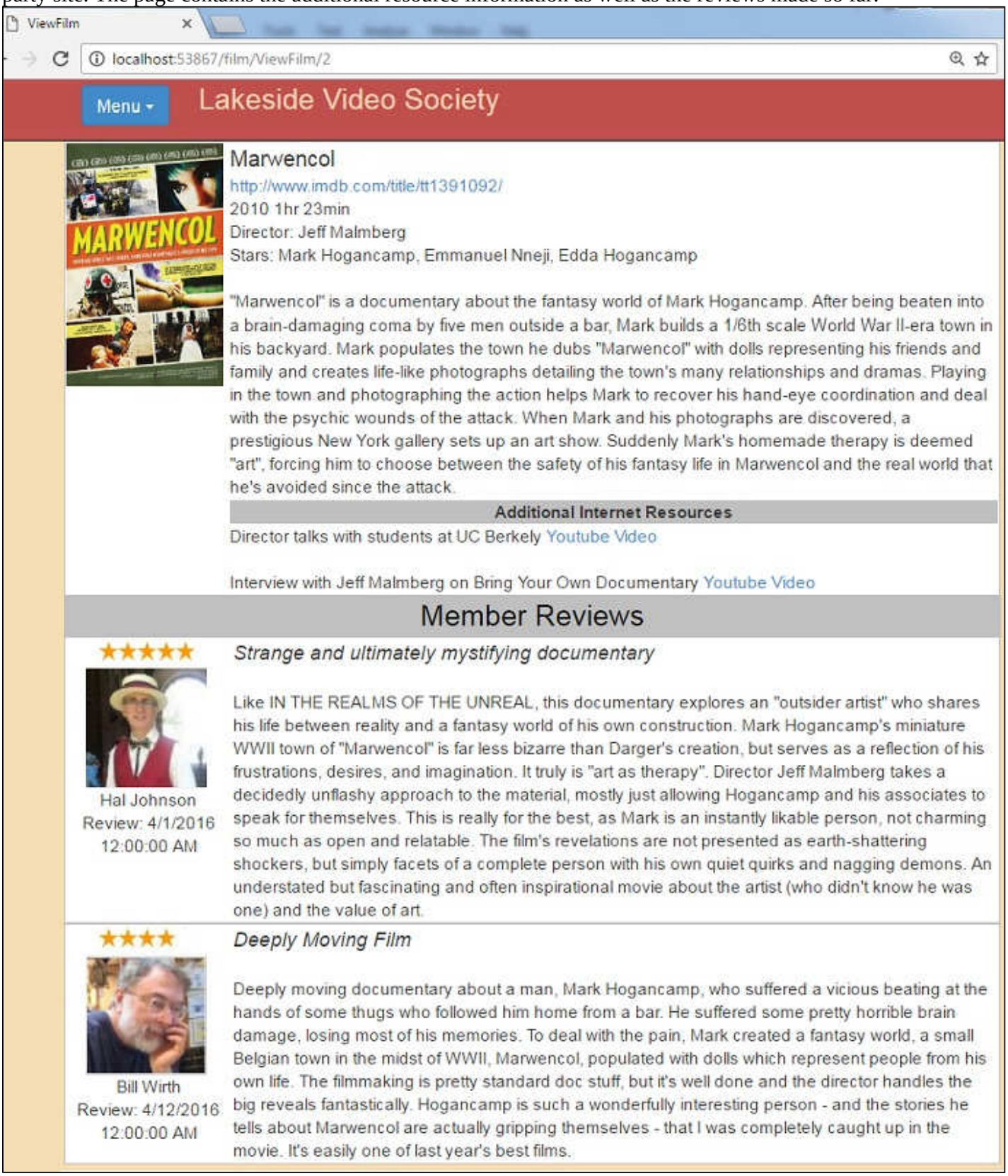


This page displays the data in the synopsis column of the Films table. This field can contain carriage return line feed character combinations. You need to use this "special" htmlhelper function in the view so that the synopsis field is properly displayed:

```
@Html.Raw(@item.Synopsis.ToString().Replace(Environment.NewLine, "<br />")
```

3. **View Film**

This page presents all the bio data on a film. It contains a "hot" link to the Internet Movie Database or another 3rd party site. The page contains the additional resource information as well as the reviews made so far.



I created 5 small jpg files for the 5 possible star ratings and called them n_star.jpg.

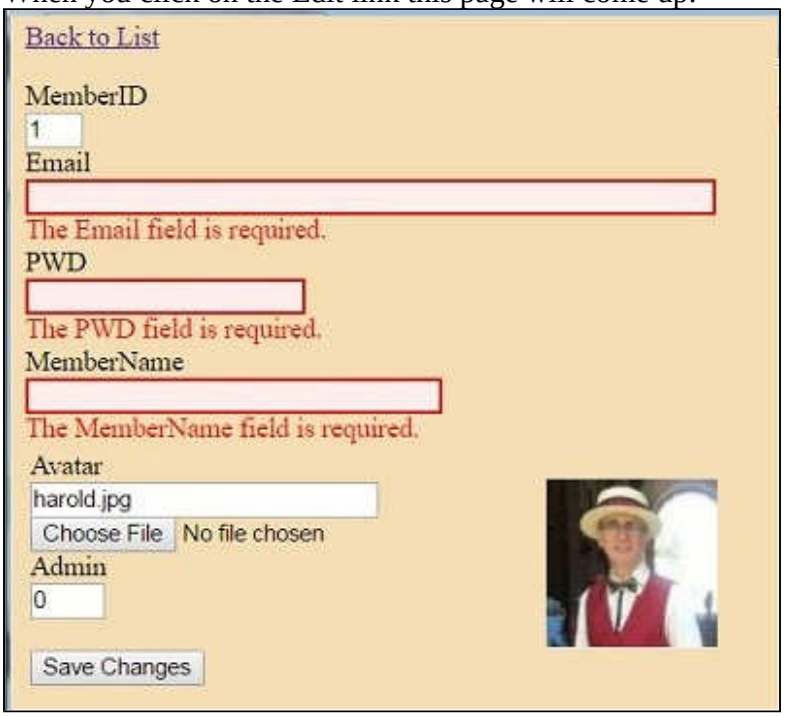
4. **Maintain Members**

This is a fairly plain vanilla gridview. I did add a link to insert a new member.
I decided not to use the masterpage and instead to put a link back to the Home page in the upper left hand corner.

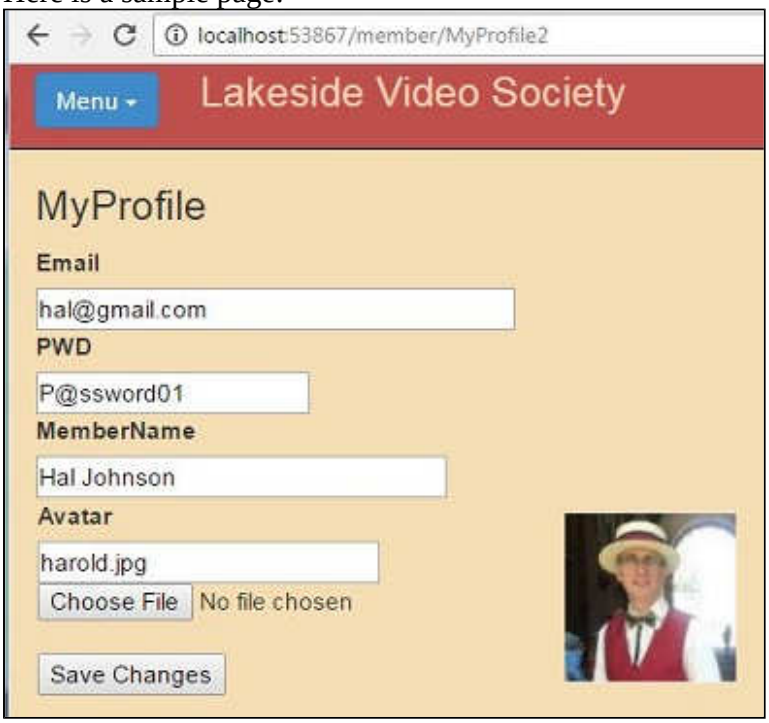
Use this code to create a new member when the Click Here link is pressed:

```
[HttpPost]
public ActionResult MemberCreate()
{
    Member mbr = new Member();
    mbr.Avatar = "noname.jpg";
    mbr.Email = "enter value";
    mbr.MemberName = "enter value";
    mbr.PWD = "P@ssword01";
    dbcon.Open();
    int intresult = Member.CUDMember(dbcon, "create", mbr);
    dbcon.Close();
}
```

When you click on the Edit link this page will come up:

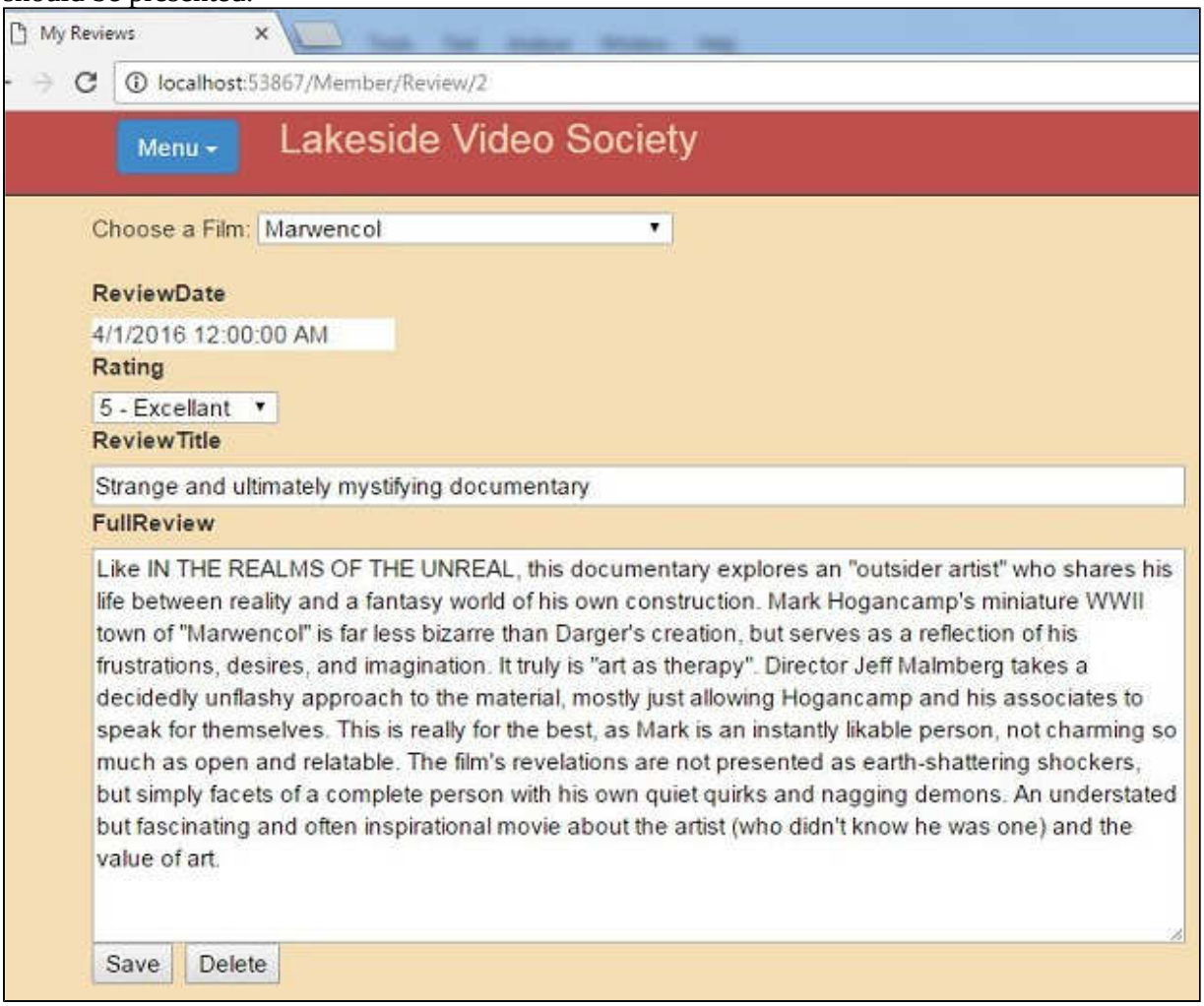


5. **MyProfile**
   Once you get the Edit Member page working you can use the same code to create the MyProfile page in the Member controller.
   Here is a sample page:



   You should note that the Memberid and admin fields are not shown on this form.

6. **Write a Review**
The page presents a dropdownlist of all the films in the database. The member selects the film and the existing review will come up with Update/Delete button. If a review does not exist then a blank form with a Create button should be presented.



You can use this code for the rating dropdownlist box:

```
@Html.DropDownListFor(m => m.review.Rating,
     new SelectList(
        new List<Object>{
            new { value = 1 , text = "1 - Awfull"},
            new { value = 2 , text = "2 - Fair"},
            new { value = 3 , text = "3 - Okay"},
            new { value = 4 , text = "4 - Very Good"},
            new { value = 5 , text = "5 - Excellant"}},
        "value", "text", 3)
)
```

7. **Error Handling**
   We will use a common error page in our application. Create a new view in the Shared folder called error.cshtml and replace the resulting razor code with this:

```
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Error</title>
</head>
<body>
    <div>
        <h2>An error has occured</h2>
        <span style="color:red">Error Message : @ViewBag.errormsg</span>
        <br /><br />
        Click <a href="~/home/index">Here</a> to return to Home page
    </div>
</body>
</html>
```

   To actually call this page we can use this code in the catch blocks:

```
try
{
  dbcon.Open();
   //statements
}
catch (Exception ex)
{
    @ViewBag.errormsg = ex.Message;
    if (dbcon != null && dbcon.State == ConnectionState.Open) dbcon.Close();
    return View("error");
}
```

8. **Security**
   Our goal is to use role based form security for our site. This will be the last part that we will implement. As a stop gap measure, we can use this action method to set the Session["memberid"] variable. Copy this code into the Index.cshtml file:

```
@{
    ViewBag.Title = "Index";
}
Welcome to the Lakeside Video Society
<br /><br />
@using (Html.BeginForm("SetMemberid", "Home"))
{
 <text>
  Current Memberid: <input type="text" name="memberid" style="width:35px"
         value="@ViewBag.memberid" />
  <br />
  <input type="submit" value="Set Memberid" />
 </text>
}@*end of beginform block*@
```

Copy this code into the HomeController.cs file:

```
// GET: Home
public ActionResult Index()
{
   if (Session["memberid"] == null)
          Session["memberid"] = 1;
   ViewBag.memberid = Session["memberid"];
   return View();
}
[HttpPost]
public ActionResult SetMemberid(int memberid)
{
   Session["memberid"] = memberid;
   return RedirectToAction("index");
}
```